## CLAIMS

We claim:

1. A data structure, comprising:

a head representing a first pointer to a first leaf node;

a tail representing a second pointer to a second leaf node; and

a body, physically adjacent to the head and to the tail, having a set of pointers pointing to contiguous empty nodes.

2. The apparatus of claim 1, wherein the nodes further comprising data of the same type.

3. The apparatus of claim 1, wherein the nodes form a sorted tree structure.

4. The apparatus of claim 1, wherein the nodes are indexed.

5. The apparatus of claim 1, wherein each leaf node comprises a number of data segments.

6. A method for rapid insertion of data segments comprising:

a sorted tree structure;

an inserting of a data segment into the tree structure; and

a redistributing of empty tree nodes by employing a data structure, which enables a more rapid insertion of the data segments.

7. The method of claim 6, wherein the data segments may be inserted in any order.

8. The method of claim 6, wherein the tree structure comprises non-leaf and leaf nodes.

9. The method of claim 6, wherein the tree nodes are indexed.

10. The method of claim 6, wherein each leaf node comprises a number of data segments.

11. The method of claim 6, wherein the redistribution data structure comprises:

a head representing a first pointer to a first leaf node;

a tail representing a second pointer to a second leaf node; and

a body, logically adjacent to the head and to the tail, having a set of pointers pointing to contiguous empty nodes.

12. The method of claim 6, wherein the redistribution process comprises the data structure traversing the tree in a first direction and a second direction.

13. The method of claim 12, wherein the first direction comprises a logical one and the second direction comprises a logical zero.

14. The method of claim 12, wherein the data structure traverses the tree by moving its head one leaf node towards its traveling direction.

15. The method of claim 12, wherein the redistribution data structure traverses the tree structure in the first direction towards non-decreasing indices.

16. The method of claim 12, wherein the redistribution data structure traverses the tree structure in the second direction towards non-increasing indices.

17. The method of claim 6, wherein the redistribution data structure traverses the tree when a data segment is inserted and two conditions are met.

18. The method of claim 17, wherein a first condition comprises a maximum threshold of filled spaces in the tree structure, and a second condition comprises a minimum threshold of filled spaces in the tree structure.

19. The method of claim 17, wherein the conditions are empirically determined.

20. The method of claim 17, wherein the redistribution data structure traverses the tree by moving one leaf node towards its traveling direction.

21. The method of claim 20, wherein the head of the redistribution data structure further comprising an empty leaf node.

22. The method of claim 21, wherein certain conditions are met and the redistribution process continues.

23. The method of claim 22, wherein the conditions are empirically calculated.

24. The method of claim 21, wherein the redistribution process halts.

25. The method of claim 24, wherein a data segment insertion restarts the redistribution process, and the traversal may continue where it was last halted.

26. The method of claim 20, wherein the head of the redistribution data structure comprises a non-empty leaf node.

27. The method of claim 26, wherein the redistribution data structure copies the contents of its head into its tail.

28. The method of claim 26, wherein the redistribution data structure travels towards non-decreasing indices.

29. The method of claim 28, wherein the tree structure updates from leaf node level to root node level.

30. The method of claim 29, wherein the contents of the head are cleared and the tail is moved a pre-calculated increment towards the traveling direction.

31. The method of claim 30, wherein the increment is empirically determined.

32. The method of claim 30, wherein certain conditions are met and the redistribution process continues.

33. The method of claim 32, wherein the conditions are empirically calculated.

34. The method of claim 30, wherein the redistribution process halts.

35. The method of claim 34, wherein a data segment insertion restarts the redistribution process, and the traversal may continue from where it was last halted.

36. The method of claim 27, wherein the redistribution data structure travels towards non-increasing indices.

37. The method of claim 36, wherein the tree structure updates between the tail and the nearest non-empty leaf node whose index is greater than the index of the tail.

38. The method of claim 37, wherein the updates further comprising changes made from leaf node level to root node level.

39. The method of claim 38, wherein the remainder of the tree structure updates from root node level to leaf node level.

40. The method of claim 39, wherein the contents of the head are cleared and the tail is moved a pre-calculated increment towards the traveling direction.

41. The method of claim 40, wherein the increment is empirically determined.

42. The method of claim 40, wherein certain conditions are met and the redistribution process continues.

43. The method of claim 42, wherein the conditions are empirically calculated.

44. The method of claim 40, wherein the redistribution process halts.

45. The method of claim 44, wherein a data segment insertion restarts the redistribution process, and the traversal may continue from where it was last halted.

46. The method of claim 6, wherein the tree structure may be reverse sorted.

47. The method of claim 6, wherein the process maintains the invariants of a sorted N-ary tree structure before and after the redistribution.

48. The method of claim 6, wherein the redistribution process maintains a consistent lookup operation on the sorted tree structure.